Appendix

A1 1D key-chest environment

In this section we provide details on the specifications, the setup, and the results on the 1D key-chest environment described in the paper. The 16 environment versions and their settings are listed in Tab. A1.

Parameters	Environment versions							
observations	POMDP	POMDP	POMDP	POMDP	POMDP	POMDP	POMDP	POMDP
n_k	1	1	1	1	3	3	3	3
p_l	0%	0%	50%	50%	0%	0%	50%	50%
n_{rnd}	0	n_{obs}	0	n_{obs}	0	n_{obs}	0	n_{obs}
time steps	32	32	99	99	48	48	148	148
Parameters	Environment versions							
observations	MDP	MDP	MDP	MDP	MDP	MDP	MDP	MDP
n_k	1	1	1	1	3	3	3	3
p_l	0%	0%	50%	50%	0%	0%	50%	50%
n_{rnd}	0	n_{obs}	0	n_{obs}	0	n_{obs}	0	n_{obs}
time steps	32	32	99	99	48	48	148	148

Table A1: Specifications of the 16 versions of the 1D key-chest environment, as illustrated in Fig. 2. Number of time steps per episode are fixed and chosen such that a random agent generates $35\%(\pm 2\%)$ episodes with a return of 1. Reward $\{0, 1\}$ is given only at the end of an episode. Training and test sets were sampled until 50% of episodes with 0 and 1 return were obtained. n_{rnd} is the number of random features with values $\{0, 1\}$ that are concatenated to the observation space, where n_{obs} is the number of features of the original observation space.

A1.1 MDP and POMDP observation space

For the fully observable MDP versions of the environment, the observation space includes information about the position, time step, number of currently held *keys*, if the agent is on position *k* or *c*, if the agent visited *c* while holding n_k *keys*, and, in case $p_l > 0$, if the current *keys* are being lost on *s*.

In the POMDP versions, the observation space only contains information if the agent is on position *k* or *c* and, in case $p_l > 0$, if the current *keys* are being lost on *s*.

Furthermore, we add $n_{rnd} = \{0, n_{obs}\}$ features that contain random values $\{0, 1\}$ to the observation space, where n_{obs} is the number of features of the original observation space. All features are binary, except for the position and time step feature, which are single floating point values.

A1.2 History compression methods

We evaluated 5 methods for history compression. Following the notation in the paper, they are:
(i) **feature-wise max-pooling:** $\boldsymbol{v}_t = \left[\boldsymbol{v}^h{}_t; \boldsymbol{v}^o{}_t\right]$; $\boldsymbol{v}^h{}_t = \max_{i=1,\ldots,(t-1)} \left(\boldsymbol{v}^o{}_i\right)$,
(ii) **feature-wise sum-pooling:** $\boldsymbol{v}_t = \left[\boldsymbol{v}^h{}_t; \boldsymbol{v}^o{}_t\right]$; $\boldsymbol{v}^h{}_t = \sum_{i=1,\ldots,(t-1)} \left(\boldsymbol{v}^o{}_i\right)$,

(iii) **fully-connected LSTM (LSTMf):** $\boldsymbol{v}_t = \left[\boldsymbol{v}^h{}_t ; \boldsymbol{v}^o{}_t\right]$; $\boldsymbol{v}^h{}_t = \mathrm{LSTMf}_{i=1,\ldots,(t-1)}\left(\boldsymbol{v}^o{}_i\right)$, with a layer of J fully-connected LSTM blocks,
(iv) **sparsely-connected LSTM (LSTMs):** $\boldsymbol{v}_t = \left[\boldsymbol{v}^h{}_t ; \boldsymbol{v}^o{}_t\right]$; $\boldsymbol{v}^h{}_t = \mathrm{LSTMs}_{i=1,\ldots,(t-1)}\left(\boldsymbol{v}^o{}_i\right)$, with a layer of J sparsely-connected LSTM blocks, where output gate, input gate, and the recurrent connections of the cell input are disabled, and
(v) **reset-max history**, as described in the main paper,
where J is the number of observation features.

A1.3 Training details

Training of all Hopfield-RUDDER, history methods, and LSTM-RUDDER versions was performed in PyTorch [3] using the Adam optimizer [2] for $10,000$ updates. Weights of the linear mappings are shared such that $m_{state} == m_{stored}$. For training of Hopfield-RUDDER, a mini-batch of 4 random samples from the training set is used as state patterns and the rest of the training set is used as stored patterns for each weight update. For LSTM-RUDDER, mini-batches of 4 random samples from the training set are used for each weight update.

For Hopfield-RUDDER, first the history features are computed from the observations. Then, if a linear mapping for Hopfield-RUDDER is used, the state and stored patterns are mapped to 16 features via a single linear layer before the association with the known episode returns is performed. The bias weights for the LSTM-based history versions and the reset-max history were initialized with random values from a normal distribution with a mean of 5.

For LSTM-RUDDER, we chose a number of 16 LSTM blocks in the network, followed by a single output layer to create the reward prediction at every time step of the episode. The LSTM loss combines the losses $L_m + 0.1\left(L_c + L_e\right)$, as described by [2], formula A275. Loss L_a was omitted, as there are no intermediate rewards given in the episodes.

A1.4 Detailed results and ablation study for history versions

In the following Fig. A1 and A2 , we show the detailed results for the different history versions.

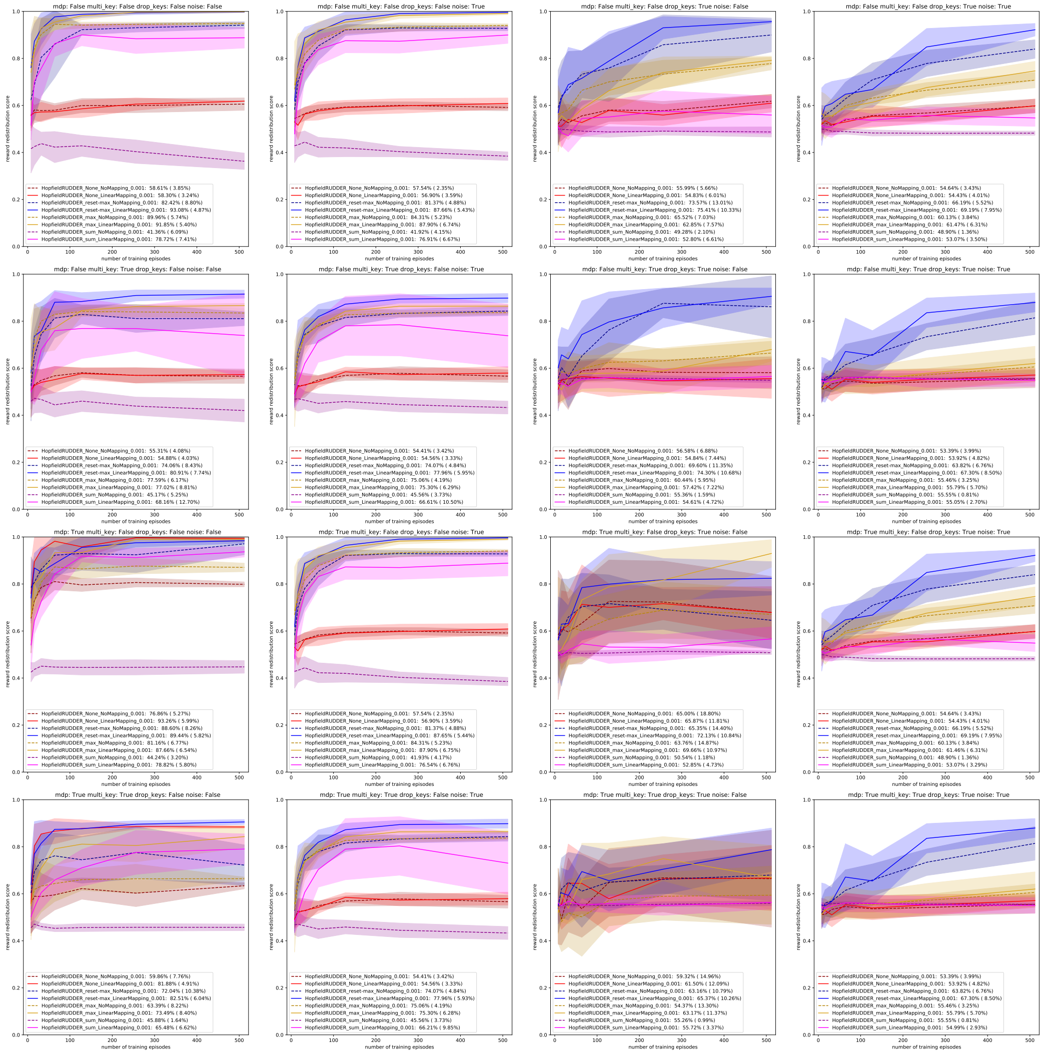

Figure A1: Comparison of Hopfield-RUDDER max-pooling, sum-pooling, and reset-max history w.r.t. reward redistribution score rr_score on different versions of the 1D key-chest environment. Results shown are the mean rr_score over 10-fold cross-validation with their corresponding standard deviations for various training set sizes.

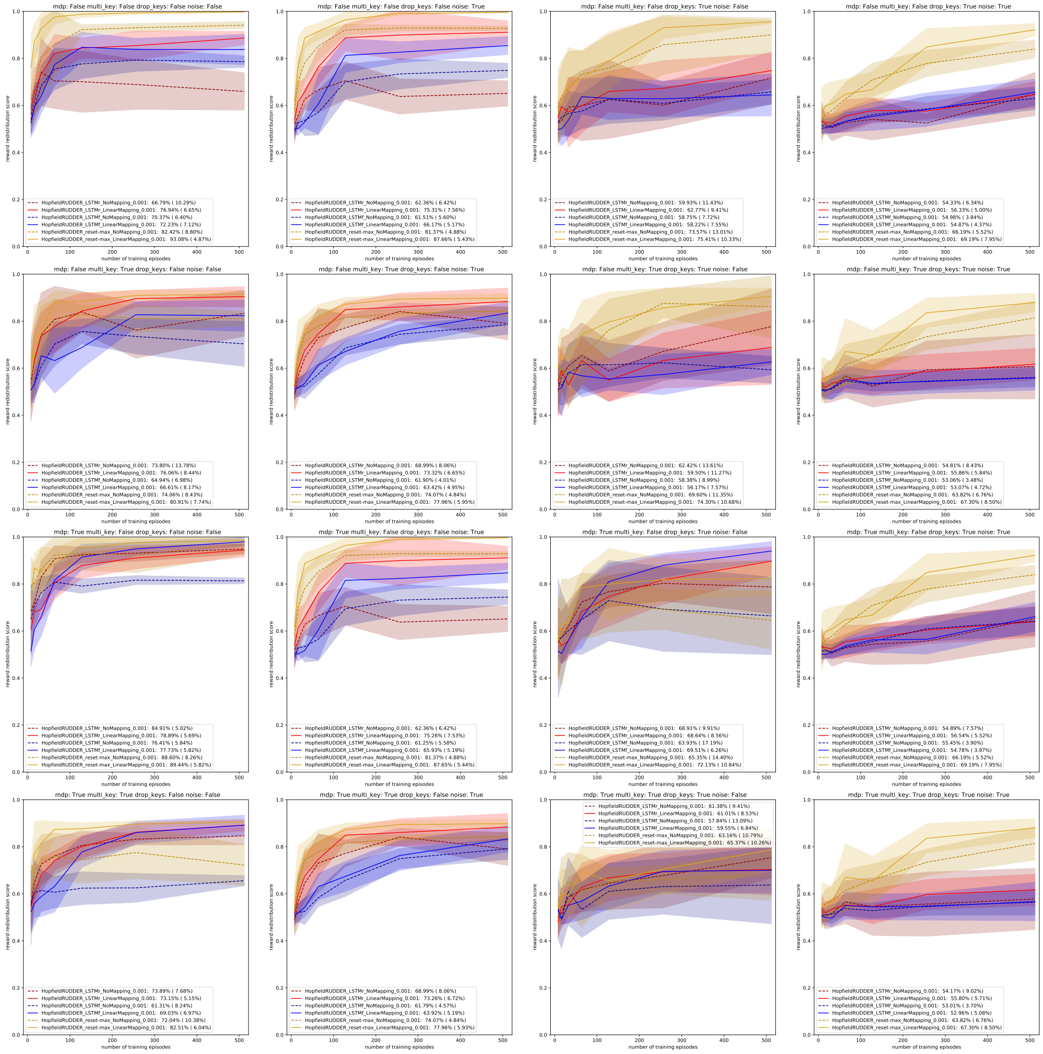

Figure A2: Comparison of Hopfield-RUDDER LSTM-based history and reset-max history w.r.t. reward redistribution score rr_score on different versions of the 1D key-chest environment. Results shown are the mean rr_score over 10-fold cross-validation with their corresponding standard deviations for various training set sizes.

A1.5 Detailed results for comparison of Hopfield-RUDDER and LSTM-RUDDER.

In the following Fig. A3, we show the detailed results for comparison of Hopfield-RUDDER with reset-max history versus the different versions of LSTM-RUDDER.

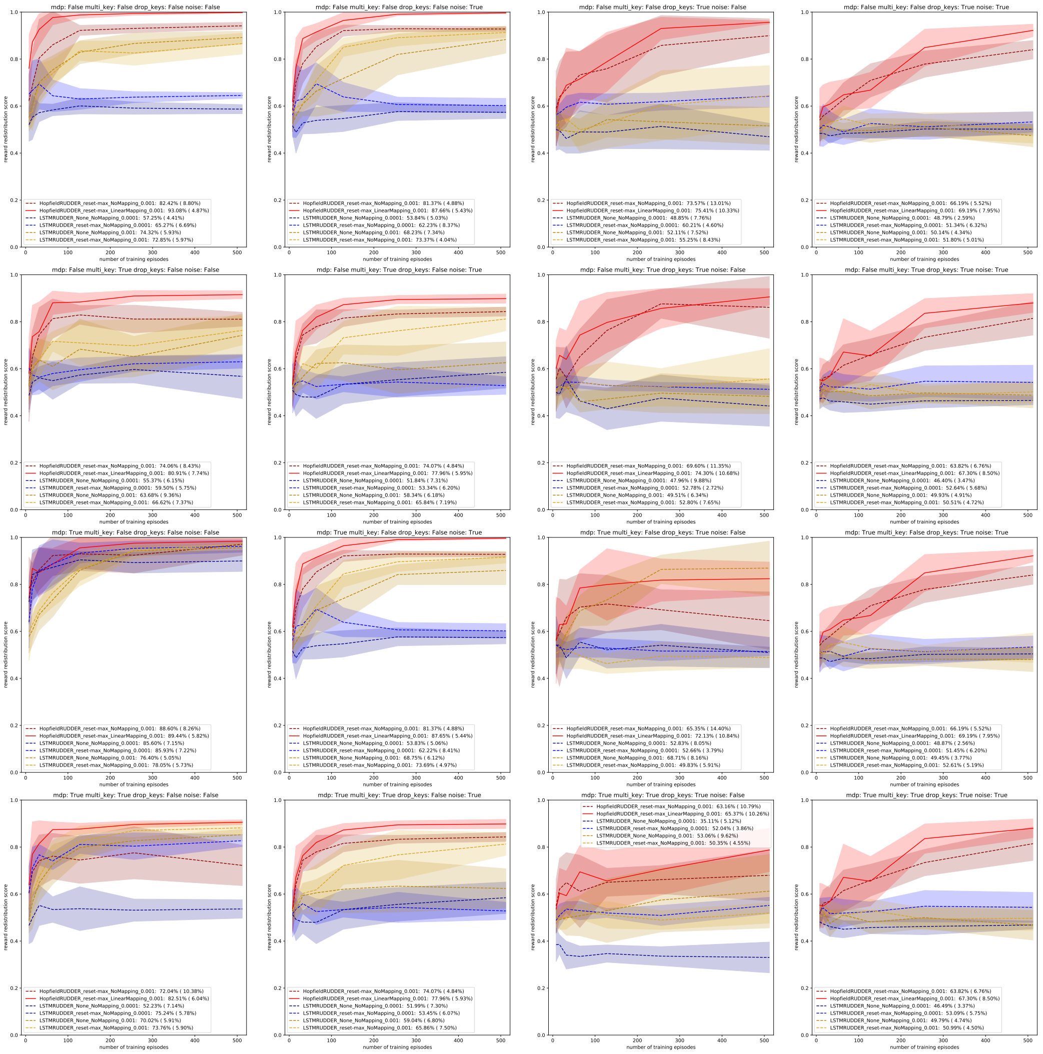

Figure A3: Comparison of Hopfield-RUDDER with reset-max history and LSTM-RUDDER versions w.r.t. reward redistribution score rr_score on different versions of the 1D key-chest environment. Results shown are the mean rr_score over 10-fold cross-validation with their corresponding standard deviations for various training set sizes.

A1.6 Detailed results for comparison of all methods.

In the following Tab. A2, we show the detailed results for comparison of all Hopfield-RUDDER and LSTM-RUDDER versions.

				Hopfield-RUDDER												LSTM-RUDDER			
History:				reset-max		LSTMf		LSTMr		max		sum		none		reset-max		none	
Mapping:				linear	none	linear	none	linear	none	linear	none	linear	none	linear	none	none	none	none	none
Learning rate:				$1e-3$	-	$1e-3$	-	$1e-3$	-	$1e-3$	-	$1e-3$	-	$1e-3$	-	$1e-4$	$1e-3$	$1e-4$	$1e-3$
Environment specs																			
	n_k	p_l	n_{rnd}																
POMDP	1	0%	0	**93.08**	82.42	72.23	70.37	76.94	66.79	91.85	89.96	78.72	41.36	58.30	58.61	65.27	72.85	57.25	74.32
				±4.87	±8.80	±7.12	±6.40	±6.65	±10.29	±5.40	±5.74	±7.41	±6.09	±3.24	±3.85	±6.69	±5.97	±4.41	±5.93
POMDP	1	0%	n_{obs}	87.66	81.37	66.17	61.51	75.31	62.36	**87.90**	84.31	76.91	41.92	56.90	57.54	62.23	73.37	53.84	68.23
				±5.43	±4.88	±5.17	±5.60	±7.56	±6.42	±6.74	±5.23	±6.67	±4.15	±3.59	±2.35	±8.37	±4.04	±5.03	±7.34
POMDP	1	50%	0	**75.41**	73.57	58.22	58.75	62.77	59.93	62.85	65.52	52.80	49.28	54.83	55.99	60.21	55.25	48.85	52.11
				±10.33	±13.01	±7.55	±7.72	±9.41	±11.43	±7.57	±7.03	±6.61	±2.10	±6.01	±5.66	±4.60	±8.43	±7.76	±7.52
POMDP	1	50%	n_{obs}	**69.19**	66.19	54.87	54.98	56.33	54.33	61.47	60.13	53.07	48.90	54.43	54.64	51.34	51.80	48.79	50.14
				±7.95	±5.52	±4.37	±3.84	±5.00	±6.34	±6.31	±3.84	±3.50	±1.36	±4.01	±3.43	±6.32	±5.01	±2.59	±4.34
POMDP	3	0%	0	**80.91**	74.06	66.61	64.94	76.06	73.80	77.02	77.59	68.16	45.17	54.88	55.31	59.50	66.62	55.37	63.68
				±7.74	±8.43	±8.17	±6.98	±8.44	±13.78	±8.81	±6.17	±12.70	±5.25	±4.03	±4.08	±5.75	±7.37	±6.15	±9.36
POMDP	3	0%	n_{obs}	**77.96**	74.07	63.42	61.90	73.32	68.99	75.30	75.06	66.61	45.56	54.56	54.41	53.34	65.84	51.84	58.34
				±5.95	±4.84	±4.95	±4.01	±6.65	±8.06	±6.29	±4.19	±10.50	±3.73	±3.33	±3.42	±6.20	±7.19	±7.31	±6.18
POMDP	3	50%	0	**74.30**	69.60	56.17	58.38	59.50	62.42	57.42	60.44	54.61	55.36	54.84	56.58	52.78	52.80	47.96	49.51
				±10.68	±11.35	±7.57	±8.99	±11.27	±13.61	±7.22	±5.95	±4.72	±1.59	±7.44	±6.88	±2.72	±7.65	±9.88	±6.34
POMDP	3	50%	n_{obs}	**67.30**	63.82	53.07	53.06	55.86	54.81	55.79	55.46	55.05	55.55	53.92	53.39	52.64	50.51	46.40	49.93
				±8.50	±6.76	±4.72	±3.48	±5.84	±8.43	±5.70	±3.25	±2.70	±0.81	±4.82	±3.99	±5.68	±4.72	±3.47	±4.91
MDP	1	0%	0	89.44	88.60	77.73	76.41	78.89	84.91	87.66	81.16	78.82	44.24	**93.26**	76.86	85.93	78.05	85.60	76.40
				±5.82	±8.26	±5.82	±5.84	±5.69	±5.02	±6.54	±6.77	±5.80	±3.20	±5.99	±5.27	±7.22	±5.73	±7.15	±5.05
MDP	1	0%	n_{obs}	87.65	81.37	65.93	61.25	75.26	62.36	**87.90**	84.31	76.54	41.93	56.90	57.54	62.22	73.69	53.83	68.75
				±5.44	±4.88	±5.19	±5.58	±7.53	±6.42	±6.75	±5.23	±6.76	±4.17	±3.59	±2.35	±8.41	±4.97	±5.06	±6.12
MDP	1	50%	0	**72.13**	65.35	69.51	63.93	68.64	68.91	69.66	63.76	52.85	50.54	65.87	65.00	52.66	49.83	52.83	68.71
				±10.84	±14.40	±6.26	±17.19	±8.56	±9.91	±10.97	±14.87	±4.73	±1.18	±11.81	±18.80	±3.79	±5.91	±8.05	±8.16
MDP	1	50%	n_{obs}	**69.19**	66.19	54.78	55.45	56.54	54.89	61.46	60.13	53.07	48.90	54.43	54.64	51.45	52.61	48.87	49.45
				±7.95	±5.52	±3.97	±3.90	±5.52	±7.57	±6.31	±3.84	±3.29	±1.36	±4.01	±3.43	±6.20	±5.19	±2.56	±3.77
MDP	3	0%	0	**82.51**	72.04	69.03	61.31	73.15	73.89	73.49	63.39	65.48	45.88	81.88	59.86	75.24	73.76	52.23	70.02
				±6.04	±10.38	±6.97	±8.24	±5.15	±7.68	±8.40	±8.22	±6.62	±1.64	±4.91	±7.76	±5.78	±5.90	±7.14	±5.91
MDP	3	0%	n_{obs}	**77.96**	74.07	63.92	61.79	73.26	68.99	75.30	75.06	66.21	45.56	54.56	54.41	53.45	65.86	51.99	59.04
				±5.93	±4.84	±5.19	±4.57	±6.72	±8.06	±6.28	±4.19	±9.85	±3.73	±3.33	±3.42	±6.07	±7.50	±7.30	±6.80
MDP	3	50%	0	**65.37**	63.16	59.55	57.84	61.01	61.38	63.17	54.37	55.72	55.26	61.50	59.32	52.04	50.35	35.11	53.06
				±10.26	±10.79	±6.84	±13.09	±8.53	±9.41	±11.37	±13.30	±3.37	±0.99	±12.09	±14.96	±3.86	±4.55	±5.12	±9.62
MDP	3	50%	n_{obs}	**67.30**	63.82	52.96	53.01	55.80	54.17	55.79	55.46	54.99	55.55	53.92	53.39	53.09	50.99	46.49	49.79
				±8.50	±6.76	±5.08	±3.70	±5.71	±9.02	±5.70	±3.25	±2.93	±0.81	±4.82	±3.99	±5.75	±4.50	±3.37	±4.74

Table A2: Comparison of different Hopfield-RUDDER and LSTM-RUDDER versions w.r.t. reward redistribution score rr_score on different versions of the 1D key-chest environment. Results shown are the mean rr_score over all training set sizes and a 10-fold cross-validation. Error bars show mean standard deviation of 10-fold cross-validation over all training set sizes. Hopfield-RUDDER with reset-max history consistently outperforms all other Hopfield-RUDDER and LSTM-RUDDER versions.

A2 Minecraft Environment

In this section we provide details on the MineRL Minecraft environment, demonstrations, training setup and additional results.

A2.1 Training details

Training of Hopfield-RUDDER was performed in PyTorch [3] using the Adam optimizer [2]. In contrast to the 1D-environment we trained the model only for 100 updates. Weights of the linear mappings are shared such that $m_{state} == m_{stored}$. For training of Hopfield-RUDDER, a mini-batch of 8 random samples from the training set is used as state patterns and the rest of the training set is used as stored patterns for each weight update.

In addition to the history compression we augment the observations with observation deltas, where we compute the delta of observation s_t and s_{t-1} and concatenate the result with the original observation. Furthermore, to reduce training time and GPU memory requirements we store only unique observations in stored patterns $\boldsymbol{Y}$.

Then, the history features are computed from the augmented observations, including actions, and the state and stored patterns are mapped to 128 features via a small neural network with 2 fully connected hidden layers with ReLU activation.

A2.2 Additional reward redistribution plots

In A4, we show additional examples for reward redistribution using Hopfield-RUDDER on the MineRL *ObtainDiamond* demonstrations. These demonstrations have not been used for training the reward redistribution model. The red vertical lines show the auxiliary sparse rewards, provided by the environment. Note that for training Hopfield-RUDDER, we only use episodic reward of 1 for successful demonstrations and 0 for unsuccessful demonstrations, without using the auxiliary rewards. As indicated by the auxiliary rewards, the reward redistribution (in blue) is able to identify sections in the episodes which are important for obtaining the diamond.

A2.3 Observation and action representation

In order to avoid hand-crafted solutions the authors of the MineRL benchmark introduced an obfuscated version of the environment and the demonstrations. In this version the inventory state and actions are encoded using an Auto Encoder. Both observations and actions are 64 dimensional vectors. The decoder of this model is not released and therefore models must learn only based on the encoded states and predict the encoded actions. We also use only the obfuscated inventory state and actions for training the Hopfield-RUDDER model. However, we use the non-obfuscated inventory states for visualizing and inspecting the reward redistribution.

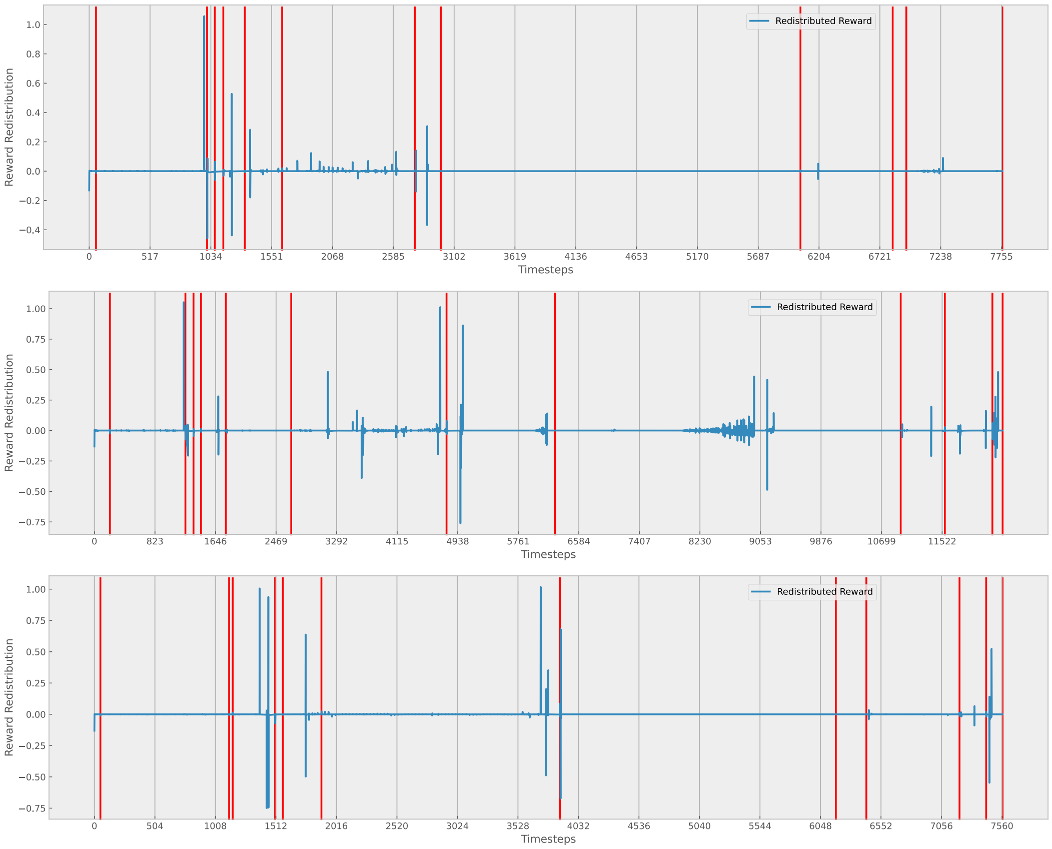

Figure A4: Reward redistribution of three demonstrations for the MineRL *ObtainDiamond* task. **Blue:** redistributed reward **Red:** sparse reward obtainable in the environment. For training we only use episodic reward of 1 for successful episodes and 0 for unsuccessful episodes.

A3 Glossary

J	.Number of features of history representation $\boldsymbol{v}^h$.
K	.Number of features of state-action pair representation $\boldsymbol{v}^o$.
$\boldsymbol{G}$	.Return values of episodes.
$\boldsymbol{U}$	.Raw state patterns (before optional mapping).
$\boldsymbol{W}$	.Trainable weights of reset gate f_{reset}.
$\boldsymbol{Y}$	.Raw stored patterns (before optional mapping).
$\boldsymbol{u}$	.Raw state pattern (before optional mapping).
$\boldsymbol{v}^h$	.Representation of the history as vector.
$\boldsymbol{v}^o$	.Representation of the state-action pair as vector.
$\boldsymbol{v}$	.Raw stored or state pattern as concatenation $[\boldsymbol{v}^o; \boldsymbol{v}^h]$.
$\boldsymbol{y}$	.Raw stored pattern (before optional mapping).
β	.Inverse temperature of $\mathrm{softmax}$ function.
ψ	.Return decomposition function.
σ	.Sigmoid function.
$\widehat{\boldsymbol{G}}$	.Estimated return values of episodes.
$\widehat{g}$	.Estimated return value of an episode.
a_l	.Action to move 1 position to the left in 1D key-chest environment.

a_r	Action to move 1 position to the right in 1D key-chest environment.
a	a_t is the action at time t.
f_{reset}	Reset gate function for resetting the history $\boldsymbol{v}^h$.
g	Return value of an episode.
m_{state}	Optional transformation function of raw $\boldsymbol{U}$.
m_{stored}	Optional transformation function of raw $\boldsymbol{Y}$.
n_k	Number of *keys* required to open *chest* in 1D key-chest environment.
n_{obs}	Number of features of observation space in 1D key-chest environment.
n_{rnd}	Number of random features in 1D key-chest environment.
p_l	Probability of losing all *keys* in 1D key-chest environment.
rr_score	Scoring function for reward redistribution for episode.
r	r_t is the reward at time t.
scr	Scoring function for reward redistribution for time step.
s	s_t is the environment state at time t.
c	Chest position of 1D key-chest environment.
k	Key position of 1D key-chest environment.
s	Mid position of 1D key-chest environment.
Hopfield-RUDDER	Novel RUDDER using MHN for return decomposition.
LSTM	Long short-term memory.
LSTM-RUDDER	Original RUDDER using LSTM for return decomposition.
LSTMf	Fully-connected LSTM.
LSTMs	Sparsely-connected LSTM.
MHN	Continuous modern Hopfield network.
RUDDER	Return decomposition for delayed rewards.

Appendix References

[1] J. A. Arjona-Medina*, M. Gillhofer*, M. Widrich*, T. Unterthiner, J. Brandstetter, and S. Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, pages 13544–13555, 2019.

[2] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *ArXiv*, 2014.

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.